PROVABLE ACCELERATION OF WIDE NEURAL NET TRAINING VIA POLYAK'S MOMENTUM

Anonymous authors
Paper under double-blind review

ABSTRACT

Incorporating a so-called "momentum" dynamic in gradient descent methods is widely used in neural net training as it has been broadly observed that, at least empirically, it often leads to significantly faster convergence. At the same time, there are very few theoretical guarantees in the literature to explain this apparent acceleration effect. In this paper we show that Polyak's momentum, in combination with over-parameterization of the model, helps achieve faster convergence in training a one-layer ReLU network on n examples. We show specifically that gradient descent with Polyak's momentum decreases the initial training error at a rate much faster than that of vanilla gradient descent. We provide a bound for a fixed sample size n, and we show that gradient descent with Polyak's momentum converges at an accelerated rate to a small error that is controllable by the number of neurons m. Prior work (Du et al., 2019b) showed that using vanilla gradient descent, and with a similar method of over-parameterization, the error decays as $(1-\kappa_n)^t$ after t iterations, where κ_n is a problem-specific parameter. Our result shows that with the appropriate choice of parameters one has a rate of $(1-\sqrt{\kappa_n})^t$. This work establishes that momentum does indeed speed up neural net training.

1 INTRODUCTION

Momentum methods are very popular for training neural networks in various applications (e.g. He et al. (2016); Vaswani et al. (2017); Krizhevsky et al. (2012)). It has been widely observed that the use of momentum helps faster training in deep learning (e.g. Sutskever et al. (2013); Hoffer et al. (2017); Loshchilov & Hutter (2019); Wilson et al. (2017); Cutkosky & Orabona (2019); Liu & Belkin (2020)). Among all the momentum methods, the most popular one seems to be Polyak's momentum (a.k.a. Heavy Ball momentum) (Polyak, 1964), which is the default choice of momentum in PyTorch and Tensorflow. [1] The success of Polyak's momentum in deep learning is widely appreciated and almost all of the recently developed adaptive gradient methods like Adam (Kingma & Ba (2015)), AMSGrad (Reddi et al. (2018)), and AdaBound (Luo et al. (2019)) adopt the use of Polyak's momentum, instead of Nesterov's momentum.

However, despite its popularity, little is known in theory about why Polyak's momentum helps to accelerate training neural networks. Even for convex optimization, smooth twice continuously differentiable functions like strongly convex quadratic problems seem to be one of the few cases that Polyak's momentum method provably achieves faster convergence than standard gradient descent (e.g. Lessard et al. (2016); Goh (2017); Ghadimi et al. (2015); Gitman et al. (2019); Loizou & Richtárik (2017; 2018); Can et al. (2019); Scieur & Pedregosa (2020); Flammarion & Bach (2015)). On the other hand, the theoretical guarantees of Adam (Kingma & Ba (2015)), AMSGrad (Reddi et al. (2018)), or AdaBound (Luo et al. (2019)) are only worse if the momentum parameter β is non-zero and the guarantees deteriorate as the momentum parameter increases, which do not show any advantage of the use of momentum (see also e.g. Alacaoglu et al. (2020)). Moreover, the convergence rates that have been established for Polyak's momentum in several related works (Gadat et al., 2016; Sun et al., 2019; Yang et al., 2018; Liu et al., 2020c) do not improve upon those for vanilla gradient descent or vanilla SGD. There are even negative cases in *convex* optimization showing that

[1]See PyTorch webpage `https://pytorch.org/docs/stable/_modules/torch/optim/sgd.html` and Tensorflow webpage `https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/SGD`.

Algorithm 1: Gradient descent with Polyak's momentum Polyak (1964) (Equivalent Version 1)

```
1: Required: Step size parameter $\eta$ and momentum parameter $\beta$.
2: Init: $w_0 \in \mathbb{R}^d$ and $m_{-1} = 0 \in \mathbb{R}^d$.
3: for $t = 0$ to $T$ do
4:     Given current iterate $w_t$, obtain gradient $\nabla L(w_t)$.
5:     Update momentum $m_t := \beta m_{t-1} + \nabla L(w_t)$.
6:     Update iterate $w_{t+1} := w_t - \eta m_t$.
7: end for
```

Algorithm 2: Gradient descent with Polyak's momentum Polyak (1964) (Equivalent Version 2)

```
1: Required: step size $\eta$ and momentum parameter $\beta$.
2: Init: $w_0 = w_{-1} \in \mathbb{R}^d$
3: for $t = 0$ to $T$ do
4:     Given current iterate $w_t$, obtain gradient $\nabla L(w_t)$.
5:     Update iterate $w_{t+1} = w_t - \eta \nabla L(w_t) + \beta(w_t - w_{t-1})$.
6: end for
```

the use of Polyak's momentum results in divergence (e.g. Lessard et al. (2016); Ghadimi et al. (2015)). Furthermore, Kidambi et al. (2018) construct a problem instance for which the momentum method under its optimal tuning is outperformed by other algorithms. A solid understanding of the empirical success of Polyak's momentum in deep learning has eluded researchers for some time.

In this paper, we provably show that Polyak's momentum helps achieve faster convergence for training a one-hidden-layer ReLU network. Over the past few years there have appeared an enormous number of works considering training a one-layer ReLU network, provably showing convergence results for vanilla (stochastic) gradient descent (e.g. Li & Liang (2018); Ji & Telgarsky (2020); Li & Yuan (2017); Du et al. (2019b;a); Allen-Zhu et al. (2019); Song & Yang (2019); Zou et al. (2019); Arora et al. (2019); Jacot et al. (2018); Lee et al. (2019); Chizat et al. (2019); Brutzkus & Globerson (2017); Tian (2017); Soltanolkotabi (2017); Bai & Lee (2020); Ghorbani et al. (2019); Li et al. (2020); Hanin & Nica (2020); Daniely (2017); Zou & Gu (2019); Dukler et al. (2020); Daniely (2020); Wei et al. (2019); Yehudai & Shamir (2020); Fang et al. (2019); Su & Yang (2019); Oymak & Soltanolkotabi (2019)) as well as for other algorithms (e.g. Zhang et al. (2019); Wu et al. (2019b); Cai et al. (2019); Wu et al. (2019a); Zhong et al. (2017); Ge et al. (2019); van den Brand et al. (2020); Lee et al. (2020)). However, we are not aware of any theoretical works that study the momentum method in neural net training except the work Krichene et al. (2020). Krichene et al. (2020) show that SGD with Polyak's momentum (a.k.a. stochastic Heavy Ball) with infinitesimal step size, i.e. $\eta \to 0$, for training a one-hidden-layer network with an infinite number of neurons, i.e. $m \to \infty$, converges to a stationary solution *asymptotically*. However, the asymptotic convergence result does not explain the faster convergence of momentum.In this paper we consider the discrete-time setting and consider nets with infinite neurons as well as nets with finitely many neurons. We provide a non-asymptotic convergence rate of Polyak's momentum, establishing a concrete improvement relative to the best-known rates for vanilla gradient descent. Our result follows the same framework as previous results, e.g. Du et al. (2019b); Arora et al. (2019); Song & Yang (2019).

We study training a one-hidden-layer ReLU neural net of the form,

$$\mathcal{N}_W(x) := \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \sigma(\langle w^{(r)}, x \rangle), \tag{1}$$

where $\sigma(z) := z \cdot \mathbb{1}\{z \geq 0\}$ is the ReLU activation, $w^{(1)}, \ldots, w^{(m)} \in \mathbb{R}^d$ are the weights of m neurons on the first layer, $a_1, \ldots, a_m \in \mathbb{R}$ are weights on the second layer, $x \in \mathbb{R}^d$ is the input, and $\mathcal{N}(x) \in \mathbb{R}$ is the output predicted on input x. Denote $W := \{w^{(r)}\}_{r=1}^m$. We consider empirical loss minimization with the squared loss,

$$L(W) := \tfrac{1}{2} \textstyle\sum_{i=1}^n \left(y_i - \mathcal{N}_W(x_i)\right)^2, \tag{2}$$

where $y_i \in \mathbb{R}$ is the label of sample x_i and n is the number of samples. Following previous works of of (Du et al., 2019b; Arora et al., 2019; Song & Yang, 2019), we define a Gram matrix $H \in \mathbb{R}^{n \times n}$

for the weights W and its expectation $\bar{H} \in \mathbb{R}^{n\times n}$ over the random draws of $w^{(r)} \sim N(0, I_d) \in \mathbb{R}^d$ whose (i,j) entries are defined as follows,

$$\begin{aligned} H(W)_{i,j} &:= \frac{1}{m}\sum_{r=1}^{m} x_i^\top x_j \mathbb{1}\{\langle w^{(r)}, x_i\rangle \geq 0 \;\&\; \langle w^{(r)}, x_j\rangle \geq 0\} \\ \bar{H}_{i,j} &:= \mathop{\mathbb{E}}_{w^{(r)}\sim N(0,I_d)}[x_i^\top x_j \mathbb{1}\{\langle w^{(r)}, x_i\rangle \geq 0 \;\&\; \langle w^{(r)}, x_j\rangle \geq 0\}]. \end{aligned} \tag{3}$$

We note that the matrix $\bar{H}$ is also called a neural tangent kernel (NTK) matrix in the literature (e.g. Jacot et al. (2018); Yang (2019); Huang et al. (2020); Bietti & Mairal (2019)). Assume that the smallest eigenvalue $\lambda := \lambda_{\min}(\bar{H})$ is strictly positive and certain conditions about the step size and the number of neurons are satisfied. Previous works of (Du et al., 2019b; Arora et al., 2019; Song & Yang, 2019) were able to show that gradient descent decreases the empirical risk (2) at a linear rate $1 - \frac{\eta\lambda}{2}$, i.e. $L(W_t) = \big(1 - \frac{\eta\lambda}{2}\big)L(W_{t-1})$. In this paper, following the same framework as (Du et al., 2019b; Arora et al., 2019; Song & Yang, 2019), we show that gradient descent with Polyak's momentum decreases the empirical risk at an accelerated linear rate $1 - \sqrt{\frac{\eta\lambda}{2}}$ to a small additive error that is controllable by the number of neurons m. [2] This shows the combined advantage of Polyak's momentum and over-parameterization. As the number of neurons m and samples n approach infinity, as considered in (Krichene et al., 2020), our analysis shows that gradient descent with Polyak's momentum converges to any arbitrarily small error at the accelerated rate.

2 Preliminaries

2.1 Polyak's momentum

Algorithm 1 and Algorithm 2 show two equivalent presentations of gradient descent with Polyak's momentum. Given the same initialization, one can show that Algorithm 1 and Algorithm 2 produce exact the same iterates during optimization. We note that for the ReLU activation, it is not differentiable at zero. So for solving (2), we replace the notion of gradient in Algorithm 1 and Algorithm 2 with subgradient $\frac{\partial L(W_t)}{\partial w_t^{(r)}} := \frac{1}{\sqrt{m}}\sum_{i=1}^n \big(\mathcal{N}_{W_t}(x_i) - y_i\big)a_r \cdot \mathbb{1}[\langle w_t^{(r)}, x_i\rangle \geq 0]x_i$ and update the neuron r as $w_{t+1}^{(r)} = w_t^{(r)} - \eta\frac{\partial L(W_t)}{\partial w_t^{(r)}} + \beta\big(w_t^{(r)} - w_{t-1}^{(r)}\big)$.

The most common example in the literature that demonstrates the advantage of Polyak's momentum over vanilla gradient descent is the strongly convex quadratic problem, $\min_{w\in\mathbb{R}^d} \frac{1}{2}w^\top A w + b^\top w$, where $A \in \mathbb{R}^{d\times d} \succ 0_d$. Applying gradient descent with Polyak's momentum (Algorithm 2) to the problem, the iterate evolves according to the following dynamics,

$$w_{t+1} - w_* = (I_d - \eta A)(w_t - w_*) + \beta(w_t - w_*) - \beta(w_{t-1} - w_*), \tag{4}$$

where I_d is the identity matrix, η is the step size, and w_* satisfies $Aw_* = b$, which is the unique minimizer of the quadratic problem. One can re-write the recursive dynamics (4) as follows,

$$\begin{bmatrix} w_{t+1} - w_* \\ w_t - w_* \end{bmatrix} = \begin{bmatrix} I_d - \eta A + \beta I_d & -\beta I_d \\ I_d & 0_d \end{bmatrix} \cdot \begin{bmatrix} w_t - w_* \\ w_{t-1} - w_* \end{bmatrix}. \tag{5}$$

A known result (see e.g. Lessard et al. (2016); Polyak (1987)) is that under an optimal tuning of the momentum parameter β. The error decays at an accelerated linear rate

$$\left\| \begin{bmatrix} w_{t+1} - w_* \\ w_t - w_* \end{bmatrix} \right\| \leq \big(1 - \sqrt{\eta\lambda_d}\big)^t \left\| \begin{bmatrix} w_t - w_* \\ w_{t-1} - w_* \end{bmatrix} \right\|, \tag{6}$$

where λ_d is the smallest eigenvalues of A. On the other hand, gradient descent only has $1 - \eta\lambda_d$ convergence rate (see e.g. Lessard et al. (2016)). In the next section, we will show that the dynamics induced in the neural network training by the momentum method is similar to the quadratic function

[2] We borrow the term "accelerated linear rate" from the convex optimization literature (Nesterov, 2013), because the result here has a resemblance to those results in convex optimization, even though the neural network training is a non-convex problem.

case here (i.e. (4) and (5)), modulo some small terms whose magnitudes are controllable. The similarity hints at why momentum helps faster neural network training.

More related works of Polyak's momentum: There is little theory work that shows any provable advantage of Polyak's momentum in non-convex optimization and deep learning. Even in convex optimization, related works make additional assumptions to show a provable advantage over standard GD or SGD. Chen & Kolar (2020) study Polyak's momentum under a growth condition. Sebbouh et al. (2020) show that SGD with Polyak's momentum outperforms vanilla SGD in smooth convex optimization when the data is interpolated. On the other hand, for smooth non-convex optimization, Wang et al. (2020) show that Polyak's momentum helps to escape saddle points faster and find a second-order stationary point faster. Yet, they also make certain assumptions regarding some statistical properties of gradient and momentum. There are also some efforts in using continuous-time techniques to analyze a broad family of momentum methods that includes Polyak's momentum (see e.g. Diakonikolas & Jordan (2019); Maddison et al. (2018)).

2.2 Assumption and prior result

As described in the introduction section, we assume that that the smallest eigenvalue of the Gram matrix $\bar{H} \in \mathbb{R}^{n \times n}$ is strictly positive, i.e. $\lambda := \lambda_{\min}(\bar{H}) > 0$. We will also denote the largest eigenvalue of the Gram matrix $\bar{H} \in \mathbb{R}^{n \times n}$ as $\lambda_{\max}(\bar{H})$. Du et al. (2019b) show that the strict positiveness assumption is indeed mild. Specifically, they show that if no two inputs are parallel, then the least eigenvalue is strictly positive. Panigrahi et al. (2020) were able to provide a quantitative lower bound under certain conditions. Following the same framework of (Du et al., 2019b), we consider that each weight vector $w^{(r)} \in \mathbb{R}^d$ is initialized according to normal distribution, $w^{(r)} \sim N(0, I_d)$, and each $a_r \in R$ is sampled from Rademacher distribution, i.e. $a_r = 1$ with probability 0.5; and $a_r = -1$ with probability 0.5. We also assume $\|x_i\| \leq 1$ for all samples i. As the previous works (e.g. Li & Liang (2018); Ji & Telgarsky (2020); Du et al. (2019b;a); Allen-Zhu et al. (2019); Song & Yang (2019); Zou et al. (2019); Arora et al. (2019); Zou & Gu (2019)), we consider only training the first layer $\{w^{(r)}\}$ and the second layer $\{a_r\}$ is fixed throughout the iterations. In the following, we denote $u_t \in \mathbb{R}^n$ whose i_{th} entry is the network prediction for sample i (i.e. $u_t[i] = \mathcal{N}_{W_t}(x_i)$) in iteration t and $y \in \mathbb{R}^n$ is the vector whose i_{th} is the label of sample i. Now let us state a prior result of gradient descent convergence due to (Du et al., 2019b).

Theorem 1. *(Theorem 4.1 in Du et al. (2019b)) Assume that $\lambda := \lambda_{\min}(\bar{H}) > 0$ and that $w_0^{(r)} \sim N(0, I_d)$ and a_r uniformly sampled from $\{-1, 1\}$. Set the number of nodes $m = \Omega(\lambda^{-4} n^6 \delta^{-3})$ and the constant step size $\eta = O(\frac{\lambda}{n^2})$. Then, with probability at least $1-\delta$ over the random initialization, vanilla gradient descent, i.e. Algorithm 1& 2 with $\beta = 0$, has*

$$\|u_t - y\|^2 \leq \left(1 - \frac{\eta\lambda}{2}\right)^t \cdot \|u_0 - y\|^2. \tag{7}$$

We note that later Song & Yang (2019) improve the network size m to $m = \Omega(\lambda^{-4} n^4 \log^3(n/\delta))$ while obtaining the same convergence rate result of vanilla gradient descent.

3 Main results

In this section, we first state the main results and provide the intuition behind the results and detailed analysis in the later subsections.

Theorem 2. *Assume that $\lambda := \lambda_{\min}(\bar{H}) > 0$ and that $w_0^{(r)} \sim N(0, I_d)$ and a_r uniformly sampled from $\{-1, 1\}$. Fix some maximum number of iterations T, set a constant step size $\eta \leq \frac{1}{2\lambda_{\max}(\bar{H})}$, fix momentum parameter $\beta = \left(1 - \sqrt{\frac{\eta\lambda}{2}}\right)^2$, and finally set a parameter $\nu > 0$ that controls the number of network nodes, chosen as $m = \Omega(\lambda^{-4} n^{4+2\nu} \log^3(n/\delta))$. Suppose that the number of samples n satisfies $\sqrt{\frac{1}{\eta\lambda}} n^{\nu} = \Omega(T)$. Then, with probability at least $1-\delta$ over the random initialization, gradient descent with Polyak's momentum (Algorithm 1 & Algorithm 2) satisfies for any $t \leq T$,*

$$\left\| \begin{bmatrix} u_t - y \\ u_{t-1} - y \end{bmatrix} \right\| \leq \left(1 - \sqrt{\frac{\eta\lambda}{2}}\right)^t \cdot \left\| \begin{bmatrix} u_0 - y \\ u_{-1} - y \end{bmatrix} \right\| + \frac{1}{2\sqrt{2} n^{\nu}} \|u_0 - y\|. \tag{8}$$

Remark 1: The step size η can be chosen as large as $\eta = 1/(2\lambda_{\max}(\bar{H}))$. Denote the condition number of the Gram matrix as $\kappa := \frac{\lambda_{\max}(\bar{H})}{\lambda_{\min}(\bar{H})}$. With the chosen step size, we have

$$\left\| \begin{bmatrix} u_t - y \\ u_{t-1} - y \end{bmatrix} \right\| \leq \left(1 - \frac{1}{2\sqrt{\kappa}}\right)^t \cdot \left\| \begin{bmatrix} u_0 - y \\ u_{-1} - y \end{bmatrix} \right\| + \frac{1}{2\sqrt{2}n^\nu} \|u_0 - y\|. \tag{9}$$

Interestingly, the rate $\left(1 - \frac{1}{2\sqrt{\kappa}}\right)$ matches that of the accelerated rate in strongly convex smooth problems (e.g. Nesterov (2013)), where the accelerated rate has an optimal dependency on the condition number $\sqrt{\kappa}$ instead of κ.

Thanks to an anonymous reviewer, we note that Wu et al. (2019b) provide an improved analysis over Du et al. (2019b), which shows that the step size η of vanilla gradient descent can be set as $\eta = \frac{1}{c_0 \lambda_{\max}(\bar{H})}$ for some quantity $c_0 > 0$, which in turns leads to a convergence rate $\left(1 - \frac{1}{c'\kappa}\right)$ for some quantity $c' > 0$. As we discussed above, this rate has a worse dependency on κ and hence is not better than what Polyak's momentum can help to achieve.

Remark 2: Note that the initialization $w_0 = w_{-1}$ ensures that $u_0 = u_{-1}$. The condition that $\sqrt{\frac{1}{\eta\lambda}} n^\nu = \Omega(T)$ can be easily satisfied when the parameter $\nu > 0$ and the number of samples n is sufficiently large. On the other hand, for the factor $\left(1 - \sqrt{\frac{\eta\lambda}{2}}\right)^T$ to be small, the number of iterations T should satisfy $T = \Omega\left(\sqrt{\frac{1}{\eta\lambda}}\right)$. Both conditions can be satisfied by appropriately setting the parameter $\nu > 0$, which in turn determines the number of neurons m.

Theorem 2 states that Polyak's momentum helps to reduce the initial error to a number $\frac{1}{2\sqrt{2}n^\nu}\|u_0 - y\|$ at the accelerated rate $1 - \sqrt{\eta\lambda/2}$, under the optimal tuning of momentum parameter $\beta = (1 - \sqrt{\eta\lambda/2})^2$. An interesting result of Theorem 2 is that it shows the benefit of over-parametrization. By increasing the number of neurons m, gradient descent with Polyak's momentum will be able to maintain the accelerated rate until it reduces the error to a smaller error. Specifically, one can control the error, i.e. the last term of (8), by specifying the parameter ν. If $\nu = 1$, then by setting the number of neurons $m = \Omega(\lambda^{-4} n^6 \log^3(n/\delta))$, Polyak's momentum can decrease the error at the accelerated rate to a number $O(\frac{1}{n})\|u_0 - y\| = O(\frac{1}{\sqrt{n}})$, where we use that the initial error satisfies $\|y - u_0\| = O(\sqrt{n})$ (see Lemma 8 in Appendix C). Similarly, if $\nu = 1.5$, then Polyak's momentum can decrease the error at the accelerated rate to a number $O(\frac{1}{n^{1.5}})\|u_0 - y\| = O(\frac{1}{n})$ under the condition that $m = \Omega(\lambda^{-4} n^7 \log^3(n/\delta))$. In other words, Polyak's momentum helps to converge at an accelerated rate **up to** an $O(\frac{1}{n^\nu}\|u_0 - y\|)$ factor. While ν can be tuned to decrease this additional factor, this is at the expense of more neurons in the hidden layer. On the other hand, vanilla GD (e.g. Du et al. (2019b); Wu et al. (2019b)) converges to an arbitrarily error linearly and does not exhibit such type of the neighborhood convergence.

When the number of samples n approaches infinity, and ν is chosen appropriately, the last term of (8) vanishes and we have

$$\left\| \begin{bmatrix} u_t - y \\ u_{t-1} - y \end{bmatrix} \right\| \leq \left(1 - \sqrt{\frac{\eta\lambda}{2}}\right)^t \cdot \left\| \begin{bmatrix} u_0 - y \\ u_{-1} - y \end{bmatrix} \right\|. \tag{10}$$

Compared to the related work (Krichene et al., 2020) that shows asymptotic convergence result of Polyak's momentum for the neural network training in the mean-field limit, our convergence rate result clearly demonstrates the advantage of Polyak's momentum. Our result also implies that over-parametrization helps acceleration in optimization. To our knowledge, in the literature, there is little theory of understanding why over-parametrization can help training a neural network faster. The only exception that we are aware of is (Arora et al., 2018), which shows that the dynamic of vanilla gradient descent for an over-parametrized objective function exhibits some momentum terms, although their message is very different from ours.

Remark 3 (iteration complexity): Let us analyze the number of iterations required to have an ϵ error. Without loss of generality, we can assume that $\epsilon = \gamma\|u_0 - y\|$ for some number $\gamma \in (0, 1)$. According to Theorem 1, we see that for the error $\|u_t - y\|$ to decrease to $\gamma\|u_0 - y\|$ for some number $\gamma \in (0, 1)$, vanilla gradient descent needs a number of iterations $T_0^g := \lceil \frac{2\log\left(\gamma\right)}{\log\left(1 - \frac{\eta\lambda}{2}\right)} \rceil$. On the other hand, gradient descent with Polyak's momentum under the optimal tuning of β takes a number of iterations $T_0^m := \lceil \frac{\log\left(\gamma/(2\sqrt{2})\right)}{\log\left(1-\sqrt{\frac{\eta\lambda}{2}}\right)} \rceil$ for the error $\|u_t - y\|$ to decrease to $\gamma\|u_0 - y\|$. To see this, let $\frac{1}{2\sqrt{2}n^{\nu}}\|u_0 - y\| = \frac{\gamma\|u_0-y\|}{2}$ on the r.h.s. of (8). Then, the number iterations for the term $\left\|\begin{bmatrix} u_t - y \\ u_{t-1} - y \end{bmatrix}\right\| \leq \left(1 - \sqrt{\frac{\eta\lambda}{2}}\right)^t \cdot \left\|\begin{bmatrix} u_0 - y \\ u_{-1} - y \end{bmatrix}\right\|$ to decrease to $\frac{\gamma\|u_0-y\|}{2}$ is $\lceil \frac{\log\left(\gamma/(2\sqrt{2})\right)}{\log\left(1-\sqrt{\frac{\eta\lambda}{2}}\right)} \rceil$. By comparing T_0^g and T_0^m, we have that

$$T_0^m \lessapprox \frac{\log\left(1 - \frac{\eta\lambda}{2}\right)}{\log\left(1 - \sqrt{\frac{\eta\lambda}{2}}\right)} T_0^g =: \theta\left(\frac{\eta\lambda}{2}\right) T_0^g, \tag{11}$$

where in the last equality, we define $\theta(\frac{\eta\lambda}{2}) := \frac{\log\left(1-\frac{\eta\lambda}{2}\right)}{\log\left(1-\sqrt{\frac{\eta\lambda}{2}}\right)}$. To analyze $\theta(\frac{\eta\lambda}{2})$, we plot the function value of $\theta(\alpha) := \frac{\log(1-\alpha)}{\log(1-\sqrt{\alpha})}$ for various $0 < \alpha < 1$ on Figure 1, which clearly shows that the $\theta(\alpha)$ decays extremely fast as α decreases. We have that $\theta(\alpha) \approxeq 0.55$ at $\alpha = 0.5$, while $\theta(\alpha) \approxeq 0.1$ at $\alpha = 10^{-2}$ and $\theta(\alpha) = 0.01$ at $\alpha = 10^{-4}$. Therefore, inequality (11) suggests that T_0^m is small compared to T_0^g. For example if $\frac{\eta\lambda}{2} = 10^{-4}$, then $T_0^m \leq 0.01 \cdot T_0^g$, which shows that Polyak's momentum makes fast progress.

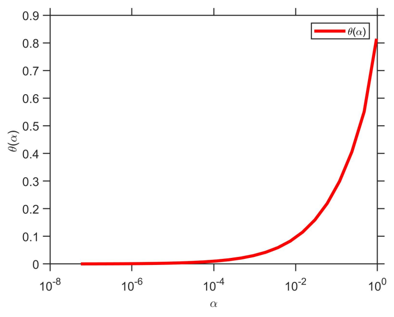

Figure 1: $\theta(\alpha) := \frac{\log(1-\alpha)}{\log(1-\sqrt{\alpha})}$ vs. α.

3.1 MORE NOTATIONS

For analysis, let us define the event $A_{ir} := \{\exists w \in \mathbb{R}^d : \|w - w_0^{(r)}\| \leq R, \mathbb{1}\{x_i^\top w_0^{(r)}\} \neq \mathbb{1}\{x_i^\top w \geq 0\}\}$, where $R > 0$ is a number to be determined later. The event A_{ir} means that there exists a $w \in \mathbb{R}^d$ which is within the R-ball centered at the initial point $w_0^{(r)}$ such that its activation pattern of sample i is different from that of $w_0^{(r)}$. We also denote a random set $S_i := \{r \in [m] : \mathbb{1}\{A_{ir}\} = 0\}$ and its complementary set $S_i^\perp := [m] \setminus S_i$. Furthermore, we denote $H_t \in \mathbb{R}^{n\times n}$ whose (i,j) entry is $H(W_t)_{i,j} = \frac{1}{m}\sum_{r=1}^m x_i^\top x_j \mathbb{1}\{\langle w_t^{(r)}, x_i\rangle \geq 0 \;\&\; \langle w_t^{(r)}, x_j\rangle \geq 0\}$. We will use the notation $\xi \in \mathbb{R}^n$ whose i_{th} entry is $\xi_t[i] := u_t[i] - y[i]$, where $u_t[i]$ is the network prediction $\mathcal{N}_t(x_i) := \mathcal{N}_{W_t}(x_i)$ at time t and $y[i]$ is the true label of sample i.

3.2 INTUITION OF THE RESULT

Applying gradient descent with Polyak's momentum to solving the objective (2) leads to the following dynamics of training errors,

$$\begin{aligned}\xi_{t+1}[i] &= \mathcal{N}_{t+1}(x_i) - y_i = \tfrac{1}{\sqrt{m}}\textstyle\sum_{r=1}^m a_r \sigma(w_{t+1}^{(r)\top} x_i) - y_i \\ &= \frac{1}{\sqrt{m}}\sum_{r=1}^m a_r \Big(w_t^{(r)\top} x_i - \tfrac{\eta}{\sqrt{m}} a_r \textstyle\sum_{j=1}^n \xi_t[j]\mathbb{1}[w_t^{(r)\top} x_j \geq 0] x_j^\top x_i \\ &\qquad + \beta(w_t^{(r)} - w_{t-1}^{(r)})^\top x_i \Big) \mathbb{1}[w_{t+1}^{(r)\top} x_i \geq 0] - y_i, \end{aligned} \tag{12}$$

where the last equality is due to the update rule of the algorithm.

Previous works like (Du et al., 2019a; Arora et al., 2019; Song & Yang, 2019) show that under certain conditions, the activation patterns of *most* of the neurons do not change, i.e. $\mathbb{1}[w_t^{(r)\top} x_j \geq 0] = \mathbb{1}[w_0^{(r)\top} x_j \geq 0]$ for all t. Now to get an intuition why momentum helps, let us for a moment assume that the patterns of *all* neurons do not change during training. Then, one can replace $\mathbb{1}[w_{t+1}^{(r)\top} x \geq 0]$

and $\mathbb{1}[w_t^{(r)\top} x \geq 0]$ with $\mathbb{1}[w_0^{(r)\top} x \geq 0]$ for any neuron r in equation (12), which leads to

$$\begin{aligned}
\xi_{t+1}[i] = \xi_t[i] &+ \beta(\xi_t[i] - \xi_{t-1}[i]) \\
&- \frac{\eta}{m} \sum_{j=1}^{n} \sum_{r=1}^{m} \xi_t[j] \cdot \mathbb{1}\left[w_0^{(r)\top} x_i \geq 0\right] \mathbb{1}\left[w_0^{(r)\top} x_j \geq 0\right] x_i^\top x_j \\
&= \xi_t[i] + \beta(\xi_t[i] - \xi_{t-1}[i]) - \eta H_0[i,:]\xi_t,
\end{aligned} \tag{13}$$

where in the last equality we use the definition of H_0 defined in Subsection 3.1. Apparently we can rewrite the above equation in a matrix form,

$$\xi_{t+1} = (I_n - \eta H_0)\xi_t + \beta(\xi_t - \xi_{t-1}). \tag{14}$$

So now we see that equation (14) and (4) are in the same form, which implies that provably showing the benefit of Polyak's momentum for the neural network training is possible. However, one has to deal with the situation that some neurons *do* change their activation patterns during training. Lemma 1 below deals with this issue.

Lemma 1. *(Dynamics of the residual error): Following the notations defined in Subsection 3.1, suppose that for all $t \in [T]$ and $r \in [m]$, $\|w_t^{(r)} - w_0^{(r)}\| \leq R$, for a number $R > 0$. Then, gradient descent with Polyak's momentum (Algorithm 1 & Algorithm 2) for (2) has*

$$\xi_{t+1} = (I_n - \eta H_t)\xi_t + \beta(\xi_t - \xi_{t-1}) + \phi_t, \tag{15}$$

where the i_{th} entry of $\phi_t \in \mathbb{R}^n$ satisfies $|\phi_t[i]| \leq \frac{2\eta\sqrt{n}|S_i^\perp|}{m}\left(\|u_t - y\| + \beta \sum_{s=0}^{t-1} \beta^{t-1-s}\|u_s - y\|\right)$.

The proof of Lemma 1 is available in Appendix A. The recursive dynamics of the residual vector ξ_t, (15), can be rewritten as

$$\begin{bmatrix} \xi_{t+1} \\ \xi_t \end{bmatrix} = \begin{bmatrix} I_n - \eta H_t + \beta I_n & -\beta I_n \\ I_n & 0 \end{bmatrix} \begin{bmatrix} \xi_t \\ \xi_{t-1} \end{bmatrix} + \begin{bmatrix} \phi_t \\ 0 \end{bmatrix}. \tag{16}$$

In the later subsection, we will show that $\|\phi_t\|$ is small and controllable. Specifically, we will use the following lemma to control $\|\phi_t\|$.

Lemma 2. *(Claim 3.12 of Song & Yang (2019)) Fix a number $R_1 \in (0,1)$. Recall that $S_i^\perp$ is a random set defined in subsection 3.1. With probability at least $1 - n \cdot \exp(-mR_1)$, we have that for all $i \in [n]$,*

$$|S_i^\perp| \leq 4mR_1.$$

A similar lemma also appears in (Du et al., 2019b). Lemma 2 says that the number of neurons whose activation patterns for a sample i could change during the execution is only a small faction of m if R_1 is a small number, i.e. $|S_i^\perp| \leq 4mR_1 \ll m$. In the later subsection, we will set $R_1 = O(\frac{\lambda}{n^{1+\nu}})$, which together with the upper-bound of $|\phi_t[i]| \leq \frac{2\eta\sqrt{n}|S_i^\perp|}{m}\left(\|u_t - y\| + \beta \sum_{s=0}^{t-1} \beta^{t-1-s}\|u_s - y\|\right)$ in Lemma 1 will allow us to control $\|\phi_t\|$.

Remark 4: We note that Liu et al. (2020b;a) establish an interesting connection between solving an over-parametrized non-linear system of equations and solving the classical linear system. They show that for smooth and twice differentiable activation, the optimization landscape of an over-parametrized network satisfies a notion called Polyak-Lokasiewicz (PL) condition (Polyak, 1963), i.e. $\frac{1}{2}\|\nabla L(w)\|^2 \geq \mu\left(L(w) - L(w_*)\right)$, where w_* is a global minimizer and $\mu > 0$. However, it is not clear if the result can be extended to ReLU, as Safran et al. (2020) show that for a one-layer ReLU network in the student-teacher setting, the PL condition does not hold after any degree of over-parametrization of the student network. Furthermore, to our knowledge, there is little theoretical result of Polyak's momentum showing an accelerated rate when an optimization landscape satisfies the PL condition but has more than one global minimum. On the other hand, for a problem that satisfies PL and has a unique global minimizer, Aujol et al. (2020) show a variant of Polyak's momentum method having an accelerated rate in a continuous-time limit. However, it is not clear if their result is applicable to our case.

3.3 DETAILED ANALYSIS

We first upper-bound the spectral norm of the matrix $\begin{bmatrix} I_n - \eta H_t + \beta I_n & -\beta I_n \\ I_n & 0 \end{bmatrix}$ on (16) as follows.

Lemma 3. *Following the setting as Theorem 2, set $m = \Omega(\lambda^{-2} n^2 \log(n/\delta))$ and the momentum parameter $\beta = \big(1 - \sqrt{\frac{\eta\lambda}{2}}\big)^2$. Suppose that the step η is chosen so that $\eta \leq \frac{1}{2\lambda_{\max}(H)}$. Then, with probability at least $1 - \delta - n^2 \exp(-m\bar{R}/10)$, for any set of weight vectors $W := \{w^{(1)}, \ldots, w^{(m)}\}$ satisfying $\|w^{(r)} - w_0^{(r)}\| \leq \bar{R} := \frac{\lambda}{8n}$ for any $r \in [m]$, it holds that*

$$\left\| \begin{bmatrix} I_n - \eta H(W) + \beta I_n & -\beta I_n \\ I_n & 0 \end{bmatrix} \right\|_2 \leq 1 - \sqrt{\tfrac{\eta\lambda}{2}}.$$

The proof of Lemma 3 is available in Appendix B. Now we are ready to prove Theorem 2.

Proof. (of Theorem 2) We will denote $R := \frac{\lambda}{64 n^{1+\nu}}$, $c_T := \max_{t \leq T} \beta_*^t (1 + \beta_* \sum_{s=0}^{t-1} \beta_*^s)$, $C := 16\sqrt{2} c_T \eta n R \|u_0 - y\|$, and $\beta_* := 1 - \sqrt{\frac{\eta\lambda}{2}} \geq \frac{1}{2}$. We will prove for all $t \in [T]$, the following inequalities hold

$$\left\| \begin{bmatrix} \xi_t \\ \xi_{t-1} \end{bmatrix} \right\| \leq \beta_*^t \cdot \left\| \begin{bmatrix} \xi_0 \\ \xi_{-1} \end{bmatrix} \right\| + \frac{C}{1 - \beta_*} \tag{17}$$

$$\|w_t^{(r)} - w_0^{(r)}\| \leq R \quad \text{and} \quad \|\phi_t\| \leq C. \tag{18}$$

The proof is by induction. For the base case $t = 0$, inequality (17) and the first inequality of (18) trivially holds. It remains to bound $\|\phi_0\|$. With probability at least $1 - n \cdot \exp(-mR)$:

$$\|\phi_0\| = \sqrt{\textstyle\sum_{i=1}^n \phi_0[i]^2} = \sqrt{\textstyle\sum_{i=1}^n \big(\frac{2\eta\sqrt{n}|S_i^\perp|}{m} \|u_0 - y\|\big)^2}$$
$$\overset{(a)}{\leq} \sqrt{\textstyle\sum_{i=1}^n (\frac{2\eta\sqrt{n} 4mR}{m})^2 \big(\|u_0 - y\|\big)^2} = 8\eta n R \|u_0 - y\| \leq C,$$

where the above inequality relies on Lemma 2, so we have that $|S_i^\perp| \leq 4mR$ for all $i \in [n]$. Now we can conclude that (17) and (18) hold for the base case 0.

Suppose that (17) and (18) hold at time $s = 0, 1, 2, \ldots, t-1$. Then,

$$\left\| \begin{bmatrix} \xi_t \\ \xi_{t-1} \end{bmatrix} \right\| \overset{(16)}{\leq} \left\| \begin{bmatrix} I_n - \eta H_{t-1} + \beta I_n & -\beta I_n \\ I_n & 0 \end{bmatrix} \right\|_2 \cdot \left\| \begin{bmatrix} \xi_{t-1} \\ \xi_{t-2} \end{bmatrix} \right\| + \left\| \begin{bmatrix} \phi_{t-1} \\ 0 \end{bmatrix} \right\|$$
$$\overset{(a)}{\leq} \beta_* \cdot \left\| \begin{bmatrix} \xi_{t-1} \\ \xi_{t-2} \end{bmatrix} \right\| + C$$
$$\overset{(b)}{\leq} \beta_*^t \cdot \left\| \begin{bmatrix} \xi_0 \\ \xi_{-1} \end{bmatrix} \right\| + \sum_{s=0}^{t-1} \beta_*^{t-1-s} C \leq \beta_*^t \cdot \left\| \begin{bmatrix} \xi_0 \\ \xi_{-1} \end{bmatrix} \right\| + \frac{C}{1 - \beta_*}, \tag{19}$$

where (a) is by Lemma 3 and the induction that $\|\phi_{t-1}\| \leq C$, (b) is by the recursive expansion of the second inequality. So (17) holds at t.

Using (19), we now show that $\|\phi_t\| \leq C$. We have that

$$\|\phi_t\| = \sqrt{\textstyle\sum_{i=1}^n \phi_t[i]^2} \leq \sqrt{\textstyle\sum_{i=1}^n \big(\frac{2\eta\sqrt{n}|S_i^\perp|}{m} \big(\|u_t - y\| + \beta \sum_{s=0}^{t-1} \beta^{t-1-s} \|u_s - y\|\big)\big)^2}$$
$$\overset{(a)}{\leq} 8\eta n R \big(\|u_t - y\| + \beta \textstyle\sum_{s=0}^{t-1} \beta^{t-1-s} \|u_s - y\|\big)$$
$$\overset{(b)}{\leq} 8\eta n R \big(\beta_*^t \sqrt{2} \|u_0 - y\| + \tfrac{C}{1-\beta_*} + \beta \textstyle\sum_{s=0}^{t-1} \beta^{t-1-s} \big(\beta_*^s \sqrt{2} \|u_0 - y\| + \tfrac{C}{1-\beta_*}\big)\big)$$
$$\overset{(c)}{\leq} 8\eta n R \big(\beta_*^t (1 + \beta_* \textstyle\sum_{s=0}^{t-1} \beta_*^s) \sqrt{2} \|u_0 - y\| + \tfrac{C}{1-\beta_*} (1 + \tfrac{\beta_*}{1-\beta_*})\big)$$
$$\overset{(d)}{\leq} C, \tag{20}$$

where (a) we use Lemma 2 so that for all $i \in [n]$, it holds that $|S_i^{\perp}| \leq 4mR$, with probability at least $1 - n \cdot \exp(-mR)$, (b) is by induction that $\|u_t - y\| \leq \beta_*^t \sqrt{2}\|u_0 - y\| + \frac{C}{1-\beta_*}$ as $u_0 = u_{-1}$, (c) uses that $\beta = \beta_*^2$, and (d) is due to that $\beta_*^t(1 + \beta_* \sum_{s=0}^{t-1} \beta_*^s) \leq c_T$ and that

$$\frac{8\eta n R c_T \sqrt{2}\|u_0 - y\|}{1 - \frac{8\eta n R}{1-\beta_*}(1 + \frac{\beta_*}{1-\beta_*})} \leq C, \tag{21}$$

which is proved as follows. Using the definition of β_* and R, we have that $1 - \frac{8\eta n R}{1-\beta_*}(1 + \frac{\beta_*}{1-\beta_*}) \geq 1 - \frac{32nR}{\lambda} \geq \frac{1}{2}$. So for (21) to hold, it suffices to have that $16\eta n R c_T \sqrt{2}\|u_0 - y\| \leq C$, which is true by the definition of C. Now we are going to show that $\|w_t^{(r)} - w_0^{(r)}\| \leq R := \frac{\lambda}{64n^{1+\nu}}$. We have that

$$\begin{aligned} \|w_t^{(r)} - w_0^{(r)}\| &\overset{(a)}{\leq} \frac{\eta\sqrt{2n}}{\sqrt{m}}\left(\frac{2}{\eta\lambda} + \frac{32nRc_T t}{\lambda}\right)\|y - u_0\| \overset{(b)}{\leq} \frac{\eta\sqrt{2n}}{\sqrt{m}}\left(\frac{4}{\eta\lambda}\right)\|y - u_0\| \\ &\overset{(c)}{=} \frac{\eta\sqrt{2n}}{\sqrt{m}}\left(\frac{4}{\eta\lambda}\right)O\left(\sqrt{n\log(m/\delta)\log^2(n/\delta)}\right) \\ &\overset{(d)}{\leq} \frac{\lambda}{64n^{1+\nu}}, \end{aligned} \tag{22}$$

where (a) is due to Lemma 4 in Appendix C, (b) is because $\frac{32nRc_T t}{\lambda} = \frac{c_T t}{2n^\nu} \leq \frac{T}{2\sqrt{2\eta\lambda}n^\nu} \leq \frac{2}{\eta\lambda}$, where we use $c_T \leq \frac{1}{\sqrt{2\eta\lambda}}$, which is shown in Lemma 9 in Appendix C, as well as the condition that $\frac{1}{\sqrt{\eta\lambda}}n^\nu = \Omega(T)$, (c) is due to Lemma 8 in Appendix C, which states that with probability at least $1 - \delta/3$, the initial error satisfies $\|y - u_0\|^2 = O(n\log(m/\delta)\log^2(n/\delta))$, and (d) is by the choice of the number of neurons $m = \Omega(\lambda^{-4}n^{4+2\nu}\log^3(n/\delta))$. So we can conclude that (18) holds at t.

Furthermore, with the choice of m, we have that $3n^2\exp(-mR/10) \leq \delta$. Finally, Lemma 9 in Appendix C shows that $\frac{C}{1-\beta_*} \leq \frac{1}{2\sqrt{2}n^\nu}\|y - u_0\|$. Thus, we have completed the proof. □

4 Conclusion

In this work, we show that Polyak's momentum helps to accelerate training a one-layer ReLU network. The insight is that the dynamic of the predictions by the neural network during training is not very different from the accelerated dynamic in solving the strongly convex quadratic functions by the same method, provided that the weights of the neural net do not move away from its initialization too much so that most of the activation patterns of the neurons remain the same during training. We note that in the literature, this is called training a neural net in the Neural Tangent Kernel (NTK) regime (Jacot et al., 2018). Recent work of (Nakkiran et al., 2019) shows that during the early stage of training, the functions that a neural net learns are some simple functions of data, and then it starts learning more complicated functions after learning the simple one. Furthermore, Hu et al. (2020) suggest that during the early stage, the network training is indeed in the NTK regime. Therefore, a possible future work is combing our results and those of (Nakkiran et al., 2019; Hu et al., 2020) to show that momentum helps to learn the simple functions faster. We hope that our work sheds light on explaining why the momentum method works well in practice.

References

Ahmet Alacaoglu, Yura Malitsky, Panayotis Mertikopoulos, and Volkan Cevher. A new regret analysis for adam-type algorithms. *ICML*, 2020.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. *ICML*, 2019.

Andersen Ang. Heavy ball method on convex quadratic problem. *Lecture note*, 2018.

Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. *ICML*, 2018.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *NeurIPS*, 2019.

Jean-Francois Aujol, Charles Dossal, and Aude Rondepierre. Convergence rates of the heavy-ball method with lojasiewicz property. *hal-02928958*, 2020.

Yu Bai and Jason D. Lee. Beyond linearization: On quadratic and higher-order approximation of wide neural networks. *ICLR*, 2020.

Alberto Bietti and Julien Mairal. On the inductive bias of neural tangent kernels. *NeurIPS*, 2019.

Alon Brutzkus and Amir Globerson. Globally optimal gradient descent for a convnet with gaussian inputs. *ICML*, 2017.

Tianle Cai, Ruiqi Gao, Jikai Hou, Siyu Chen, Dong Wang, Di He, Zhihua Zhang, and Liwei Wang. A gram-gauss-newton method learning overparameterized deep neural networks for regression problems. *arXiv.org:1905.11675*, 2019.

Bugra Can, Mert Gürbüzbalaban, and Lingjiong Zhu. Accelerated linear convergence of stochastic momentum methods in wasserstein distances. *ICML*, 2019.

You-Lin Chen and Mladen Kolar. Understanding accelerated stochastic gradient descent via the growth condition. *arXiv:2006.06782*, 2020.

Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *NeurIPS*, 2019.

Ashok Cutkosky and Francesco Orabona. Momentum-based variance reduction in non-convex sgd. *NeurIPS*, 2019.

Amit Daniely. Sgd learns the conjugate kernel class of the network. *NeurIPS*, 2017.

Amit Daniely. Memorizing gaussians with no over-parameterizaion via gradient decent on neural networks. *arXiv:1909.11837*, 2020.

Jelena Diakonikolas and Michael I. Jordan. Generalized momentum-based methods: A hamiltonian perspective. *arXiv:1906.00436*, 2019.

Simon S Du, Jason D Lee, Haochuan Li, Liwei Wang, , and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. *ICML*, 2019a.

Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *ICLR*, 2019b.

Yonatan Dukler, Quanquan Gu, and Guido Montufar. Optimization theory for relu neural networks trained with normalization layers. *ICML*, 2020.

Cong Fang, Hanze Dong, and Tong Zhang. Over parameterized two-level neural networks can learn near optimal feature representations. *arXiv:1910.11508*, 2019.

Nicolas Flammarion and Francis Bach. From averaging to acceleration, there is only a step-size. *COLT*, 2015.

Sébastien Gadat, Fabien Panloup, and Sofiane Saadane. Stochastic heavy ball. *arXiv:1609.04228*, 2016.

Rong Ge, Rohith Kuditipudi, Zhize Li, and Xiang Wang. Learning two-layer neural networks with symmetric inputs. *ICLR*, 2019.

Euhanna Ghadimi, Hamid Reza Feyzmahdavian, and Mikael Johansson. Global convergence of the heavy-ball method for convex optimization. *ECC*, 2015.

Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, , and Andrea Montanari. Linearized two-layers neural networks in high dimension. *arXiv:1904.12191*, 2019.

Igor Gitman, Hunter Lang, Pengchuan Zhang, and Lin Xiao. Understanding the role of momentum in stochastic gradient methods. *NeurIPS*, 2019.

Gabriel Goh. Why momentum really works. *Distill*, 2017.

Boris Hanin and Mihai Nica. Finite depth and width corrections to the neural tangent kernel. *ICLR*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. *NIPS*, 2017.

Wei Hu, Lechao Xiao, Ben Adlam, and Jeffrey Pennington. The surprising simplicity of the early-time learning dynamics of neural networks. *NeurIPS*, 2020.

Kaixuan Huang, Yuqing Wang, Molei Tao, and Tuo Zhao. Why do deep residual networks generalize better than deep feedforward networks? — a neural tangent kernel perspective. *arXiv:2002.06262*, 2020.

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *NeurIPS*, 2018.

Ziwei Ji and Matus Telgarsky. Polylogarithmic width suffices for gradient descent to achieve arbitrarily small test error with shallow relu networks. *ICLR*, 2020.

Rahul Kidambi, Praneeth Netrapalli, Prateek Jain, and Sham M. Kakade. On the insufficiency of existing momentum schemes for stochastic optimization. *ICLR*, 2018.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

Walid Krichene, Kenneth F. Caluyay, and Abhishek Halder. Global convergence of second-order dynamics in two-layer neural networks. *arXiv:2006.07867*, 2020.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, 2012.

Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *NeurIPS*, 2019.

Jason D. Lee, Ruoqi Shen, Zhao Song, Mengdi Wang, and Zheng Yu. Generalized leverage score sampling for neural networks. *arXiv:2009.09829*, 2020.

Laurent Lessard, Benjamin Recht, and Andrew Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 2016.

Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. *NeurIPS*, 2018.

Yuanzhi Li and Yang Yuan. Convergence analysis of two-layer neural networks with relu activation. *NeurIPS*, 2017.

Yuanzhi Li, Tengyu Ma, and Hongyang Zhang. Learning over-parametrized two-layer relu neural networks beyond ntk. *COLT*, 2020.

Chaoyue Liu and Mikhail Belkin. Accelerating sgd with momentum for over-parameterized learning. *ICLR*, 2020.

Chaoyue Liu, Libin Zhu, and Mikhail Belkin. On the linearity of large non-linear models: when and why the tangent kernel is constant. *arXiv:2010.01092*, 2020a.

Chaoyue Liu, Libin Zhu, and Mikhail Belkin. Toward a theory of optimization for over-parameterized systems of non-linear equations: the lessons of deep learning. *arXiv:2003.00307*, 2020b.

Yanli Liu, Yuan Gao, and Wotao Yin. An improved analysis of stochastic gradient descent with momentum. *arXiv:2007.07989*, 2020c.

Nicolas Loizou and Peter Richtárik. Momentum and stochastic momentum for stochastic gradient, newton, proximal point and subspace descent methods. *arXiv:1712.09677*, 2017.

Nicolas Loizou and Peter Richtárik. Accelerated gossip via stochastic heavy ball method. *Allerton*, 2018.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *ICLR*, 2019.

Liangchen Luo, Yuanhao Xiong, Yan Liu, and Xu Sun. Adaptive gradient methods with dynamic bound of learning rate. *ICLR*, 2019.

Chris J. Maddison, Daniel Paulin, Yee Whye Teh, Brendan O'Donoghue, and Arnaud Doucet. Hamiltonian descent methods. *arXiv:1809.05042*, 2018.

Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, Tristan Yang, Benjamin L. Edelman, Fred Zhang, and Boaz Barak. Sgd on neural networks learns functions of increasing complexity. *NeurIPS*, 2019.

Yurii Nesterov. Introductory lectures on convex optimization: a basic course. *Springer*, 2013.

Samet Oymak and Mahdi Soltanolkotabi. Towards moderate overparameterization: global convergence guarantees for training shallow neural networks. *arXiv:1902.04674*, 2019.

Abhishek Panigrahi, Abhishek Shetty, and Navin Goyal. Effect of activation functions on the training of overparametrized neural nets. *ICLR*, 2020.

Boris Polyak. Gradient methods for minimizing functionals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 1963.

Boris T. Polyak. Introduction to optimization. *Optimization Software*, 1987.

B.T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 1964.

Benjamin Recht. Lyapunov analysis and the heavy ball method. *Lecture note*, 2018.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *ICLR*, 2018.

Itay Safran, Gilad Yehudai, and Ohad Shamir. The effects of mild over-parameterization on the optimization landscape of shallow relu neural networks. *arXiv:2006.01005*, 2020.

Michael Saunders. Notes on first-order methods for minimizing smooth functions. *Lecture note*, 2018.

Damien Scieur and Fabian Pedregosa. Universal average-case optimality of polyak momentum. *ICML*, 2020.

Othmane Sebbouh, Robert M. Gower, and Aaron Defazio. On the convergence of the stochastic heavy ball method. *arXiv:2006.07867*, 2020.

Mahdi Soltanolkotabi. Learning relus via gradient descent. *NeurIPS*, 2017.

Zhao Song and Xin Yang. Quadratic suffices for over-parametrization via matrix chernoff bound. *arXiv:1906.03593*, 2019.

Lili Su and Pengkun Yang. On learning over-parameterized neural networks: A functional approximation perspective. *NeurIPS*, 2019.

Tao Sun, Penghang Yin, Dongsheng Li, Chun Huang, Lei Guan, and Hao Jiang. Non-ergodic convergence analysis of heavy-ball algorithms. *AAAI*, 2019.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. *ICML*, 2013.

Yuandong Tian. An analytical formula of population gradient for two-layered relu network and its applications in convergence and critical point analysis. *ICML*, 2017.

Jan van den Brand, Binghui Peng, Zhao Song, and Omri Weinstein. Training (overparametrized) neural networks in near-linear time. *arXiv:2006.11648*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, and et al. Attention is all you need. *NIPS*, 2017.

Jun-Kun Wang, Chi-Heng Lin, and Jacob Abernethy. Escaping saddle points faster with stochastic momentum. *ICLR*, 2020.

Colin Wei, Jason D. Lee, Qiang Liu, and Tengyu Ma. Regularization matters: Generalization and optimization of neural nets v.s. their induced kernel. *NeurIPS*, 2019.

Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, , and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *NIPS*, 2017.

Shanshan Wu, Alexandros G Dimakis, and Sujay Sanghavi. Learning distributions generated by one-layer relu networks. *NeurIPS*, 2019a.

Xiaoxia Wu, Simon S Du, and Rachel Ward. Global convergence of adaptive gradient methods for an over-parameterized neural network. *arXiv:1902.07111*, 2019b.

Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv:1902.04760*, 2019.

Tianbao Yang, Qihang Lin, and Zhe Li. Unified convergence analysis of stochastic momentum methods for convex and non-convex optimization. *IJCAI*, 2018.

Gilad Yehudai and Ohad Shamir. Learning a single neuron with gradient methods. *COLT*, 2020.

Guodong Zhang, James Martens, and Roger B Grosse. Fast convergence of natural gradient descent for over-parameterized neural networks. *NeurIPS*, 2019.

Kai Zhong, Zhao Song, Prateek Jain, Peter L. Bartlett, and Inderjit S. Dhillon. Recovery guarantees for one-hidden-layer neural networks. *ICML*, 2017.

Difan Zou and Quanquan Gu. An improved analysis of training over-parameterized deep neural networks. *NeurIPS*, 2019.

Difan Zou, Yuan Cao, Dongruo Zhou, and Quanquan Gu. Stochastic gradient descent optimizes overparameterized deep relu networks. *Machine Learning, Springer*, 2019.

A Proof of Lemma 1

Lemma 1: (*Dynamics of the residual error*): *Following the notations defined in Subsection 3.1, suppose that for all $t \in [T]$ and $r \in [m]$, $\|w_t^{(r)} - w_0^{(r)}\| \leq R$, for a number $R > 0$. Then, gradient descent with Polyak's momentum (Algorithm 1 & Algorithm 2) for (2) has*

$$\xi_{t+1} = (I_n - \eta H_t)\xi_t + \beta(\xi_t - \xi_{t-1}) + \phi_t, \tag{23}$$

where the i_{th} entry of $\phi_t \in \mathbb{R}^n$ satisfies $|\phi_t[i]| \leq \frac{2\eta\sqrt{n}|S_i^\perp|}{m}\left(\|u_t - y\| + \beta \sum_{s=0}^{t-1} \beta^{t-1-s}\|u_s - y\|\right)$.

Proof. For each sample i, we will divide the contribution to $\mathcal{N}(x_i)$ into two groups.

$$\begin{aligned}\mathcal{N}(x_i) &= \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \sigma(\langle w^{(r)}, x_i \rangle) \\ &= \frac{1}{\sqrt{m}} \sum_{r \in S_i} a_r \sigma(\langle w^{(r)}, x_i \rangle) + \frac{1}{\sqrt{m}} \sum_{r \in S_i^\perp} a_r \sigma(\langle w^{(r)}, x_i \rangle).\end{aligned} \tag{24}$$

To continue, let us recall some notations; the subgradient with respect to $w^{(r)} \in \mathbb{R}^d$ is

$$\frac{\partial L(W)}{\partial w^{(r)}} := \frac{1}{\sqrt{m}} \sum_{i=1}^{n} \big(\mathcal{N}(x_i) - y_i\big) a_r x_i \mathbb{1}\{\langle w^{(r)}, x\rangle \geq 0\}. \tag{25}$$

and the Gram matrix H_t whose (i,j) element is

$$H_t[i,j] := \frac{1}{m} x_i^\top x_j \sum_{r=1}^{m} \mathbb{1}\{\langle w_t^{(r)}, x_i\rangle \geq 0 \ \&\ \langle w_t^{(r)}, x_j\rangle \geq 0\}. \tag{26}$$

Let us also denote

$$H_t^\perp[i,j] := \frac{1}{m} x_i^\top x_j \sum_{r \in S_i^\perp} \mathbb{1}\{\langle w_t^{(r)}, x_i\rangle \geq 0 \ \&\ \langle w_t^{(r)}, x_j\rangle \geq 0\}. \tag{27}$$

We have that

$$\begin{aligned}\xi_{t+1}[i] &= \mathcal{N}_{t+1}(x_i) - y_i \\ &\overset{(24)}{=} \underbrace{\frac{1}{\sqrt{m}} \sum_{r \in S_i} a_r \sigma(\langle w_{t+1}^{(r)}, x_i\rangle)}_{\text{first term}} + \frac{1}{\sqrt{m}} \sum_{r \in S_i^\perp} a_r \sigma(\langle w_{t+1}^{(r)}, x_i\rangle) - y_i \end{aligned} \tag{28}$$

For the first term above, we have that

$$\begin{aligned}
&\underbrace{\frac{1}{\sqrt{m}} \sum_{r \in S_i} a_r \sigma(\langle w_{t+1}^{(r)}, x_i\rangle)}_{\text{first term}} = \frac{1}{\sqrt{m}} \sum_{r \in S_i} a_r \sigma(\langle w_t^{(r)} - \eta \frac{\partial L(W_t)}{\partial w_t^{(r)}} + \beta(w_t^{(r)} - w_{t-1}^{(r)}), x_i\rangle) \\
&= \frac{1}{\sqrt{m}} \sum_{r \in S_i} a_r \langle w_t^{(r)} - \eta \frac{\partial L(W_t)}{\partial w_t^{(r)}} + \beta(w_t^{(r)} - w_{t-1}^{(r)}), x_i\rangle \cdot \mathbb{1}\{\langle w_{t+1}^{(r)}, x_i\rangle \geq 0\} \\
&\overset{(a)}{=} \frac{1}{\sqrt{m}} \sum_{r \in S_i} a_r \langle w_t^{(r)}, x_i\rangle \cdot \mathbb{1}\{\langle w_t^{(r)}, x_i\rangle \geq 0\} + \frac{\beta}{\sqrt{m}} \sum_{r \in S_i} a_r \langle w_t^{(r)}, x_i\rangle \cdot \mathbb{1}\{\langle w_t^{(r)}, x_i\rangle \geq 0\} \\
&\quad - \frac{\beta}{\sqrt{m}} \sum_{r \in S_i} a_r \langle w_{t-1}^{(r)}, x_i\rangle \cdot \mathbb{1}\{\langle w_{t-1}^{(r)}, x_i\rangle \geq 0\} - \eta \frac{1}{\sqrt{m}} \sum_{r \in S_i} a_r \langle \frac{\partial L(W_t)}{\partial w_t^{(r)}}, x_i\rangle \mathbb{1}\{\langle w_t^{(r)}, x_i\rangle \geq 0\} \\
&= \mathcal{N}_t(x_i) + \beta\big(\mathcal{N}_t(x_i) - \mathcal{N}_{t-1}(x_i)\big) - \frac{1}{\sqrt{m}} \sum_{r \in S_i^\perp} a_r \langle w_t^{(r)}, x_i\rangle \mathbb{1}\{\langle w_t^{(r)}, x_i\rangle \geq 0\} \\
&\quad - \frac{\beta}{\sqrt{m}} \sum_{r \in S_i^\perp} a_r \langle w_t^{(r)}, x_i\rangle \mathbb{1}\{\langle w_t^{(r)}, x_i\rangle \geq 0\} + \frac{\beta}{\sqrt{m}} \sum_{r \in S_i^\perp} a_r \langle w_{t-1}^{(r)}, x_i\rangle \mathbb{1}\{\langle w_{t-1}^{(r)}, x_i\rangle \geq 0\}\big) \\
&\quad \underbrace{- \eta \frac{1}{\sqrt{m}} \sum_{r \in S_i} a_r \langle \frac{\partial L(W_t)}{\partial w_t^{(r)}}, x_i\rangle \mathbb{1}\{\langle w_t^{(r)}, x_i\rangle \geq 0\}}_{\text{last term}}
\end{aligned} \tag{29}$$

where (a) uses that for $r \in S_i$, $\mathbb{1}\{\langle w_{t+1}^{(r)}, x_i\rangle \geq 0\} = \mathbb{1}\{\langle w_t^{(r)}, x_i\rangle \geq 0\} = \mathbb{1}\{\langle w_{t-1}^{(r)}, x_i\rangle \geq 0\}$ as the neurons in S_i do not change their activation patterns. We can further bound (29) as

$$\begin{aligned}
&\overset{(b)}{=} \mathcal{N}_t(x_i) + \beta\big(\mathcal{N}_t(x_i) - \mathcal{N}_{t-1}(x_i)\big) - \eta \sum_{j=1}^{n} \big(\mathcal{N}_t(x_j) - y_j\big) H(W_t)_{i,j} \\
&\quad - \frac{\eta}{m} \sum_{j=1}^{n} x_i^\top x_j (\mathcal{N}_t(x_j) - y_j) \sum_{r \in S_i^\perp} \mathbb{1}\{\langle w_t^{(r)}, x_i\rangle \geq 0 \ \&\ \langle w_t^{(r)}, x_j\rangle \geq 0\} \\
&\quad - \frac{1}{\sqrt{m}} \sum_{r \in S_i^\perp} a_r \langle w_t^{(r)}, x_i\rangle \mathbb{1}\{\langle w_t^{(r)}, x_i\rangle \geq 0\} - \frac{\beta}{\sqrt{m}} \sum_{r \in S_i^\perp} a_r \langle w_t^{(r)}, x_i\rangle \mathbb{1}\{\langle w_t^{(r)}, x_i\rangle \geq 0\} \\
&\quad + \frac{\beta}{\sqrt{m}} \sum_{r \in S_i^\perp} a_r \langle w_{t-1}^{(r)}, x_i\rangle \mathbb{1}\{\langle w_{t-1}^{(r)}, x_i\rangle \geq 0\}\big),
\end{aligned} \tag{30}$$

where (b) is due to that

$$\begin{aligned}
&\underbrace{\frac{1}{\sqrt{m}} \textstyle\sum_{r \in S_i} a_r \langle \frac{\partial L(W_t)}{\partial w_t^{(r)}}, x_i \rangle \mathbb{1}\{\langle w_t^{(r)}, x_i \rangle \geq 0\}}_{\text{last term}} \\
=& \frac{1}{m} \sum_{j=1}^{n} x_i^\top x_j (\mathcal{N}_t(x_j) - y_j) \sum_{r \in S_i} \mathbb{1}\{\langle w_t^{(r)}, x_i \rangle \geq 0 \;\&\; \langle w_t^{(r)}, x_j \rangle \geq 0\} \\
=& \sum_{j=1}^{n} \big(\mathcal{N}_t(x_j) - y_j\big) H(W_t)_{i,j} - \frac{1}{m} \sum_{j=1}^{n} x_i^\top x_j (\mathcal{N}_t(x_j) - y_j) \sum_{r \in S_i^\perp} \mathbb{1}\{\langle w_t^{(r)}, x_i \rangle \geq 0 \;\&\; \langle w_t^{(r)}, x_j \rangle \geq 0\}.
\end{aligned} \tag{31}$$

Combining (28) and (30), we have that

$$\begin{aligned}
\xi_{t+1}[i] = \xi_t[i] &+ \beta\big(\xi_t[i] - \xi_{t-1}[i]\big) - \eta \sum_{j=1}^{n} H_t[i,j] \xi_t[j] \\
&- \frac{\eta}{m} \sum_{j=1}^{n} x_i^\top x_j (\mathcal{N}_t(x_j) - y_j) \sum_{r \in S_i^\perp} \mathbb{1}\{\langle w_t^{(r)}, x_i \rangle \geq 0 \;\&\; \langle w_t^{(r)}, x_j \rangle \geq 0\} \\
&+ \frac{1}{\sqrt{m}} \sum_{r \in S_i^\perp} a_r \sigma(\langle w_{t+1}^{(r)}, x_i \rangle) - a_r \sigma(\langle w_t^{(r)}, x_i \rangle) - \beta a_r \sigma(\langle w_t^{(r)}, x_i \rangle) + \beta a_r \sigma(\langle w_{t-1}^{(r)}, x_i \rangle).
\end{aligned} \tag{32}$$

So we can write the above into a matrix form.

$$\xi_{t+1} = (I_n - \eta H_t) \xi_t + \beta(\xi_t - \xi_{t-1}) + \phi_t, \tag{33}$$

where the i element of $\phi_t \in \mathbb{R}^n$ is defined as

$$\begin{aligned}
\phi_t[i] = &- \frac{\eta}{m} \sum_{j=1}^{n} x_i^\top x_j (\mathcal{N}_t(x_j) - y_j) \sum_{r \in S_i^\perp} \mathbb{1}\{\langle w_t^{(r)}, x_i \rangle \geq 0 \;\&\; \langle w_t^{(r)}, x_j \rangle \geq 0\} \\
&+ \frac{1}{\sqrt{m}} \sum_{r \in S_i^\perp} \big\{ a_r \sigma(\langle w_{t+1}^{(r)}, x_i \rangle) - a_r \sigma(\langle w_t^{(r)}, x_i \rangle) - \beta a_r \sigma(\langle w_t^{(r)}, x_i \rangle) + \beta a_r \sigma(\langle w_{t-1}^{(r)}, x_i \rangle) \big\}.
\end{aligned} \tag{34}$$

Now let us bound $\phi_t[i]$ as follows.

$$\begin{aligned}
\phi_t[i] &= -\frac{\eta}{m}\sum_{j=1}^{n} x_i^\top x_j(\mathcal{N}_t(x_j)-y_j)\sum_{r\in S_i^\perp}\mathbb{1}\{\langle w_t^{(r)},x_i\rangle \geq 0 \ \&\ \langle w_t^{(r)},x_j\rangle \geq 0\} \\
&\quad + \frac{1}{\sqrt{m}}\sum_{r\in S_i^\perp}\left\{a_r\sigma(\langle w_{t+1}^{(r)},x_i\rangle) - a_r\sigma(\langle w_t^{(r)},x_i\rangle) - \beta a_r\sigma(\langle w_t^{(r)},x_i\rangle) + \beta a_r\sigma(\langle w_{t-1}^{(r)},x_i\rangle)\right\} \\
&\overset{(a)}{\leq} \frac{\eta\sqrt{n}|S_i^\perp|}{m}\|u_t-y\| + \frac{1}{\sqrt{m}}\sum_{r\in S_i^\perp}\left(\|w_{t+1}^{(r)}-w_t^{(r)}\| + \beta\|w_t^{(r)}-w_{t-1}^{(r)}\|\right) \\
&\overset{(b)}{=} \frac{\eta\sqrt{n}|S_i^\perp|}{m}\|u_t-y\| + \frac{\eta}{\sqrt{m}}\sum_{r\in S_i^\perp}\left(\|\sum_{s=0}^{t}\beta^{t-s}\frac{\partial L(W_s)}{\partial w_s^{(r)}}\| + \beta\|\sum_{s=0}^{t-1}\beta^{t-1-s}\frac{\partial L(W_s)}{\partial w_s^{(r)}}\|\right) \\
&\overset{(c)}{\leq} \frac{\eta\sqrt{n}|S_i^\perp|}{m}\|u_t-y\| + \frac{\eta}{\sqrt{m}}\sum_{r\in S_i^\perp}\left(\sum_{s=0}^{t}\beta^{t-s}\|\frac{\partial L(W_s)}{\partial w_s^{(r)}}\| + \beta\sum_{s=0}^{t-1}\beta^{t-1-s}\|\frac{\partial L(W_s)}{\partial w_s^{(r)}}\|\right) \\
&\overset{(d)}{\leq} \frac{\eta\sqrt{n}|S_i^\perp|}{m}\|u_t-y\| + \frac{\eta\sqrt{n}|S_i^\perp|}{m}\left(\sum_{s=0}^{t}\beta^{t-s}\|u_s-y\| + \beta\sum_{s=0}^{t-1}\beta^{t-1-s}\|u_s-y\|\right) \\
&= \frac{2\eta\sqrt{n}|S_i^\perp|}{m}\left(\|u_t-y\| + \beta\sum_{s=0}^{t-1}\beta^{t-1-s}\|u_s-y\|\right),
\end{aligned} \tag{35}$$

where (a) is because $-\frac{\eta}{m}\sum_{j=1}^{n}x_i^\top x_j(\mathcal{N}_t(x_j)-y_j)\sum_{r\in S_i^\perp}\mathbb{1}\{\langle w_t^{(r)},x_i\rangle \geq 0 \ \&\ \langle w_t^{(r)},x_j\rangle \geq 0\} \leq \frac{\eta|S_i^\perp|}{m}\sum_{j=1}^{n}|\mathcal{N}_t(x_j)-y_j| \leq \frac{\eta\sqrt{n}|S_i^\perp|}{m}\|u_t-y\|$, and that $\sigma(\cdot)$ is 1-Lipschitz so that

$$\begin{aligned}
\frac{1}{\sqrt{m}}\sum_{r\in S_i^\perp}\left(a_r\sigma(\langle w_{t+1}^{(r)},x_i\rangle) - a_r\sigma(\langle w_t^{(r)},x_i\rangle)\right) &\leq \frac{1}{\sqrt{m}}\sum_{r\in S_i^\perp}|\langle w_{t+1}^{(r)},x_i\rangle - \langle w_t^{(r)},x_i\rangle| \\
\leq \frac{1}{\sqrt{m}}\sum_{r\in S_i^\perp}\|w_{t+1}^{(r)}-w_t^{(r)}\|\|x_i\| &\leq \frac{1}{\sqrt{m}}\sum_{r\in S_i^\perp}\|w_{t+1}^{(r)}-w_t^{(r)}\|,
\end{aligned}$$

similarly, $\frac{-\beta}{\sqrt{m}}\sum_{r\in S_i^\perp}\left(a_r\sigma(\langle w_t^{(r)},x_i\rangle) - a_r\sigma(\langle w_{t-1}^{(r)},x_i\rangle)\right) \leq \beta\frac{1}{\sqrt{m}}\sum_{r\in S_i^\perp}\|w_t^{(r)}-w_{t-1}^{(r)}\|$, (b) is by the update rule (Algorithm 1), (c) is by Jensen's inequality, (d) is because $|\frac{\partial L(W_s)}{\partial w_s^{(r)}}| = |\frac{1}{\sqrt{m}}\sum_{i=1}^{n}\left(u_s[i]-y_i\right)a_r x_i\mathbb{1}\{x^\top w_t^{(r)} \geq 0\}| \leq \frac{\sqrt{n}}{m}\|u_s-y\|$.

□

B PROOF OF LEMMA 3

Lemma 3: *Following the setting as Theorem 2, set* $m = \Omega(\lambda^{-2}n^2\log(n/\delta))$ *and the momentum parameter* $\beta = \left(1-\sqrt{\frac{\eta\lambda}{2}}\right)^2$. *Suppose that the step* η *is chosen so that* $\eta \leq \frac{1}{2\lambda_{\max}(\bar{H})}$. *Then, with probability at least* $1-\delta-n^2\cdot\exp(-m\bar{R}/10)$, *for any set of weight vectors* $W := \{w^{(1)},\dots,w^{(m)}\}$ *satisfying* $\|w^{(r)}-w_0^{(r)}\| \leq \bar{R} := \frac{\lambda}{8n}$ *for any* $r \in [m]$, *it holds that*

$$\|\begin{bmatrix} I_n-\eta H(W)+\beta I_n & -\beta I_n \\ I_n & 0 \end{bmatrix}\|_2 \leq 1-\sqrt{\frac{\eta\lambda}{2}}.$$

Proof. Denote $M := \|\begin{bmatrix} I_n-\eta H(W)+\beta I_n & -\beta I_n \\ I_n & 0 \end{bmatrix}$ on (16). Denote $\lambda^{(1)} \geq \lambda^{(2)} \geq \cdots \geq \lambda^{(n)}$ eigenvalues of H in a decreasing order. To obtain the spectral norm M, it suffices to consider the spectral norm of the sub-matrix $M_k := \| \begin{bmatrix} 1 - \eta\lambda^{(k)} + \beta & -\beta \\ 1 & 0 \end{bmatrix} \|_2 \in \mathbb{R}^{2\times 2}$ for each $k \in [n]$, and one will have $M = \max_k\{M_k\}$, which is a known technique in the literature (see e.g. Saunders (2018); Ang (2018); Recht (2018)). The eigenvalues of the 2×2 matrix is given by the roots of $p(k) := z^2 - (1 + \beta - \eta\lambda^{(k)})z + \beta$. It can be shown that when β is at least $(1 - \sqrt{\eta\lambda^{(k)}})^2$, then the magnitude of the roots of each $p(k)$ are at most $\sqrt{\beta}$ (see Lemma 5 in Appendix C). It remains to bound $\lambda^{(n)}$. We have that

$$\lambda^{(n)} := \lambda_{\min}(H(W)) \geq \lambda_{\min}(H_0) - \|H_0 - H(W)\|_F \geq \tfrac{3}{4}\lambda - \tfrac{1}{4}\lambda \geq \tfrac{\lambda}{2}, \tag{36}$$

where in the second to last inequality, we use Lemma 6 in Appendix C, which states that with probability at least $1 - \delta$, the smallest eigenvalue satisfies $\lambda_{\min}(H_0) \geq \frac{3}{4}\lambda$ under the condition of the number of neurons, i.e. $m = \Omega(\lambda^{-2}n^2 \log(n/\delta))$, and we also use Lemma 7 in Appendix C, which shows that if $\|w^{(r)} - w_0^{(r)}\| \leq \bar{R} := \frac{\lambda}{8n}$ for all $r \in [m]$, then with probability at least $1 - n^2 \exp(-m\frac{\bar{R}}{10})$, it holds that $\|H_0 - H(W)\|_F \leq 2n\bar{R} = 2n\frac{\lambda}{8n} = \frac{\lambda}{4}$.

By using that $\|H_0 - H(W)\|_F \leq \frac{\lambda}{4}$, we also have

$$\begin{aligned} \lambda^{(1)} := \lambda_{\max}(H(W)) &\leq \lambda_{\max}(H_0) + \|H_0 - H(W)\|_F \leq \lambda_{\max}(H_0) + \frac{\lambda}{4} \\ &\leq \lambda_{\max}(\bar{H}) + \|H_0 - \bar{H}\|_F + \frac{\lambda}{4} \leq \lambda_{\max}(\bar{H}) + \frac{\lambda}{2}, \end{aligned} \tag{37}$$

where in the last inequality we use Lemma 6. Therefore, for $\eta \leq \frac{1}{2\lambda_{\max}(\bar{H})}$, we have $\eta\lambda^{(k)} \leq 1$ for any $k \in [n]$. Consequently, we have $\left(1 - \sqrt{\eta\lambda^{(n)}}\right)^2 \overset{(36)}{\leq} \left(1 - \sqrt{\frac{\eta\lambda}{2}}\right)^2 = \beta$, which in turn implies that $M = \max_k\{M_k\} \leq \sqrt{\beta} = 1 - \sqrt{\frac{\eta\lambda}{2}}$. □

C SOME SUPPORTING LEMMAS

We will also need the following lemma, which shows that the iterate during the execution of the algorithm is not far away from its initialization. Similar results appear in the previous works (e.g. Li & Liang (2018); Ji & Telgarsky (2020); Du et al. (2019b;a); Allen-Zhu et al. (2019); Song & Yang (2019); Zou et al. (2019); Arora et al. (2019); Zou & Gu (2019)).

Lemma 4. *Following the setting as Theorem 2, if for any $s \leq t$, the residual dynamics satisfies $\| \begin{bmatrix} \xi_s \\ \xi_{s-1} \end{bmatrix} \| \leq \beta_*^s \cdot \| \begin{bmatrix} \xi_0 \\ \xi_{-1} \end{bmatrix} \| + \frac{C}{1-\beta_*}$, where $C := 16\sqrt{2}\eta n R c_T \|u_0 - y\|$, then we have that*

$$\|w_{t+1}^{(r)} - w_0^{(r)}\| \leq \tfrac{\eta\sqrt{2n}}{\sqrt{m}} C_{\beta_*,t} \|y - u_0\|,$$

for all $r \in [m]$, where $C_{\beta_*,t} := \frac{1}{(1-\beta_*)^2} + \frac{16\eta n R c_T (t+1)}{(1-\beta_*)^2} \leq \frac{2}{\eta\lambda} + \frac{32 n R c_T (t+1)}{\lambda}$.

Proof.

$$
\begin{aligned}
\|w_{t+1}^{(r)} - w_0^{(r)}\| &\overset{(a)}{\leq} \eta \sum_{s=0}^{t} \|m_s^{(r)}\| \overset{(b)}{=} \eta \sum_{s=0}^{t} \| \sum_{\tau=0}^{s} \beta^{s-\tau} \frac{\partial L(W_\tau)}{\partial w_\tau^{(r)}} \| \leq \eta \sum_{s=0}^{t} \sum_{\tau=0}^{s} \beta^{s-\tau} \| \frac{\partial L(W_\tau)}{\partial w_\tau^{(r)}} \| \\
&\overset{(c)}{\leq} \eta \sum_{s=0}^{t} \sum_{\tau=0}^{s} \beta^{s-\tau} \frac{\sqrt{n}}{\sqrt{m}} \|y - u_\tau\| \\
&\overset{(d)}{\leq} \eta \sum_{s=0}^{t} \sum_{\tau=0}^{s} \beta^{s-\tau} \frac{\sqrt{n}}{\sqrt{m}} (\beta_*^\tau \sqrt{2} \|y - u_0\| + \frac{C}{1-\beta_*}) \\
&\overset{(e)}{\leq} \frac{\eta\sqrt{2n}}{\sqrt{m}} \sum_{s=0}^{t} \frac{\beta_*^s}{1-\beta_*} \|y - u_0\| + \frac{\eta\sqrt{n}C}{\sqrt{m}(1-\beta_*)} \sum_{s=0}^{t} \sum_{\tau=0}^{s} \beta_*^{2(s-\tau)} \\
&\overset{(f)}{\leq} \frac{\eta\sqrt{2n}}{\sqrt{m}} \frac{1}{(1-\beta_*)^2} \|y - u_0\| + \frac{\eta\sqrt{n}\big(16\sqrt{2}\eta n R c_T \|u_0 - y\|\big)(t+1)}{\sqrt{m}(1-\beta_*)(1-\beta_*^2)} \\
&\overset{(g)}{=} \frac{\eta\sqrt{2n}}{\sqrt{m}} C_{\beta_*,t} \|y - u_0\|,
\end{aligned}
\tag{38}
$$

where (a), (b) is by the update rule of momentum, which is $w_{t+1}^{(r)} - w_t^{(r)} = -\eta m_t^{(r)}$, where $m_t^{(r)} := \sum_{s=0}^{t} \beta^{t-s} \frac{\partial L(W_s)}{\partial w_s^{(r)}}$, (c) is because $\| \frac{\partial L(W_s)}{\partial w_s^{(r)}} \| = \| \sum_{i=1}^{n} (y_i - u_s[i]) \frac{1}{\sqrt{m}} a_r x_i \cdot \mathbb{1}\{\langle w_s^{(r)}, x\rangle \geq 0\} \| \leq \frac{1}{\sqrt{m}} \sum_{i=1}^{n} |y_i - u_s[i]| \leq \frac{\sqrt{n}}{\sqrt{m}} \|y - u_s\|_2$, (d) is due to the assumption that $\| \begin{bmatrix} \xi_s \\ \xi_{s-1} \end{bmatrix} \| \leq \beta_*^s \cdot \| \begin{bmatrix} \xi_0 \\ \xi_{-1} \end{bmatrix} \| + \frac{C}{1-\beta_*}$, (e) is because that $\beta = \beta_*^2$, (f) is by $\sum_{k=1}^{\infty} k\theta^k = \frac{\theta}{(1-\theta)^2}$ for any $\theta \in [0,1)$, and (g) we denote $C_{\beta_*,t} := \frac{1}{(1-\beta_*)^2} + \frac{16\eta n R c_T (t+1)}{(1-\beta_*)^2}$. Finally, by using that $\beta_* := (1 - \sqrt{\frac{\eta\lambda}{2}})$, we have that $C_{\beta_*,t} \leq \frac{2}{\eta\lambda} + \frac{32 n R c_T (t+1)}{\lambda}$. The proof is completed.

□

Lemma 5. *The roots of the characteristics equation, $z^2 - (1 + \beta - \eta\lambda_k)z + \beta = 0$, have magnitude $|z| \leq \sqrt{\beta}$, if $\beta \geq (1 - \sqrt{\eta\lambda_k})^2$.*

Proof. The roots of $z^2 - (1 + \beta - \eta\lambda_k)z + \beta = 0$ are $z = \frac{1+\beta-\eta\lambda_k \pm \sqrt{(1+\beta-\eta\lambda_k)^2 - 4\beta}}{2}$. The magnitude of the roots are the same when the roots are imaginary, which is $|z| = \sqrt{\frac{(1+\beta-\eta\lambda_k)^2 + 4\beta - (1+\beta-\eta\lambda_k)^2}{4}} = \sqrt{\beta}$.

Simple calculation shows that $(1 + \beta - \eta\lambda_k)^2 - 4\beta \leq 0$ if $\beta \geq (1 - \sqrt{\eta\lambda_k})^2$.

□

Lemma 6. *(Lemma 3.1 in Du et al. (2019b) and Song & Yang (2019)) Denote $\lambda := \lambda_{\min}(\bar{H})$. Set $m = \Omega(\lambda^{-2} n^2 \log(n/\delta))$. Suppose that $\tilde{w}_1, \ldots, \tilde{w}_m$ are i.i.d. generated $N(0, I_d)$. Then, it holds that*

$$\|H(\tilde{W}) - \bar{H}\|_F \leq \tfrac{\lambda}{4} \textit{ and } \lambda_{\min}\big(H(\tilde{W})\big) \geq \tfrac{3}{4}\lambda,$$

with probability at least $1 - \delta$.

Lemma 7. *(Lemma 3.2 in Song & Yang (2019)) Fix a number $R_0 \in (0,1)$. Suppose that $\tilde{w}_1, \ldots, \tilde{w}_m$ are i.i.d. generated $N(0, I_d)$. Then, for any set of weight vectors $w_1, \ldots, w_m \in \mathbb{R}^d$ that satisfy for any $r \in [m]$, $\|\tilde{w}_r - w_r\|_2 \leq R_0$, it holds that*

$$\|H(\tilde{W}) - H(W)\|_F < 2nR_0,$$

with probability at least $1 - n^2 \cdot \exp(-mR_0/10)$,

Lemma 8. *(Claim 3.10 in Song & Yang (2019)) Assume that $w_0^{(r)} \sim N(0, I_d)$ and a_r uniformly sampled from $\{-1, 1\}$. For $0 < \delta < 1$, we have that*

$$\|y - u_0\|^2 = O(n \log(m/\delta) \log^2(n/\delta)),$$

with probability at least $1 - \delta$.

Lemma 9. *Denote $R := \frac{\lambda}{64n^{1+\nu}}$, $c_T := \max_{t \le T} \beta_*^t(1 + \beta_* \sum_{s=0}^{t-1} \beta_*^s)$, $C := 16\sqrt{2}c_T \eta n R \|u_0 - y\|$, and $\beta_* := 1 - \sqrt{\frac{\eta\lambda}{2}}$. Then, $c_T \le \frac{1}{4\beta_* \sqrt{\frac{\eta\lambda}{2}}}$ and $\frac{C}{1-\beta_*} \le \frac{1}{4\sqrt{2}\beta_* n^\nu}\|y - u_0\|$. Furthermore, if $\eta\lambda \le \frac{1}{2}$, then $\beta_* \ge \frac{1}{2}$; and consequently,*

$$c_T \le \frac{1}{\sqrt{2\eta\lambda}}$$
$$\frac{C}{1-\beta_*} \le \frac{1}{2\sqrt{2}n^\nu}\|y - u_0\|.$$

Proof. We have that

$$\frac{C}{1-\beta_*} = \frac{\frac{\sqrt{2}}{4}\eta\lambda c_T \|u_0 - y\|}{n^\nu \sqrt{\frac{\eta\lambda}{2}}} = \frac{c_T\sqrt{\eta\lambda}}{2n^\nu}\|u_0 - y\|. \tag{39}$$

So it remains to bound $c_T := \max_{t \le T} \beta_*^t(1 + \beta_* \sum_{s=0}^{t-1} \beta_*^s)$. Let us denote $x := \beta_*^t$. Note that $x \le 1$. Consider maximize $h(x) := x(1 + \beta_* \frac{1-x}{1-\beta_*}) = \frac{x}{1-\beta_*} - \frac{\beta_*}{1-\beta_*}x^2$. The derivative is $\nabla h(x) = \frac{1}{1-\beta_*} - \frac{2\beta_*}{1-\beta_*}x$. So the maximal value is at $x = \frac{1}{2\beta_*}$ and we have that $h(\frac{1}{2\beta_*}) = \frac{1}{4\beta_*(1-\beta_*)} = \frac{1}{4(1-\sqrt{\frac{\eta\lambda}{2}})\sqrt{\frac{\eta\lambda}{2}}}$. So we have that $c_T \le \frac{1}{4\beta_*\sqrt{\frac{\eta\lambda}{2}}}$. Substituting it back to (39), we have that

$$\frac{C}{1-\beta_*} \le \frac{1}{4\sqrt{2}\beta_* n^\nu}\|y - u_0\|. \tag{40}$$

□

D EXPERIMENT

In this section, we report a proof-of-concept experiment. We sample $n = 5$ points from the normal distribution, and then scale the size to the unit norm. We generate the labels uniformly random from $\{1, -1\}$. We let $m = 1000$ and $d = 10$. We compare vanilla GD and gradient descent with Polyak's momentum. We use the empirical Gram matrix at the initialization as an estimate of $\bar{H}$. Denote $\hat{\lambda}_{\max} := \lambda_{\max}(H_0)$ and $\hat{\lambda}_{\min} := \lambda_{\min}(H_0)$. Then, for gradient descent with Polyak's momentum, we set the step size $\eta = 1/\left(10\hat{\lambda}_{\max}\right)$ and set the momentum parameter $\beta = (1 - \sqrt{\eta\hat{\lambda}_{\min}})^2$. For gradient descent, we set the same step size.

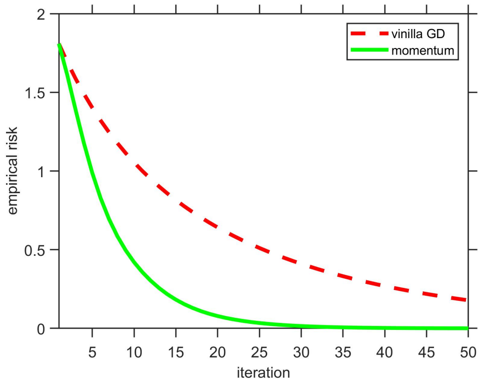

Figure 2: Empirical risk $L(W_t)$ vs. iteration t. Polyak's momentum accelerates the optimization process.

We also report the percentiles of pattern changes over iterations. Specifically, we report the quantity

$$\frac{\sum_{i=1}^{n}\sum_{r=1}^{m} \mathbb{1}\{\text{sign}(x_i^\top w_t^{(r)}) \neq \text{sign}(x_i^\top w_0^{(r)})\}}{mn},$$

as there are mn patterns. For gradient descent with Polyak's momentum, the percentiles of pattern changes is approximately 0.76%; while for vanilla gradient descent, the percentiles of pattern changes is 0.55%.